images are first filtered, a binary mask is generated, and that the contour(s) of each object is detected in each image based on the binary mask generated.

[0058] In some examples, the object pose detector **203** may be further configured to detect a location within the image representation of each object being held by the user, in the obtained images. The image representation may correspond to one or more contours detected as representing the object being held. The location may correspond to a generally central point or region with respect to the contour detected for each object.

[0059] The object detector may be configured to determine a respective location within the image representations of each object by calculating an image moment for the pixels within the contours detected for each non-luminous object. This may be useful where, for example, the objects are symmetrical, such as oranges, and so the central points or regions (e.g. centre of mass) can be calculated relatively easily. The moments( ) function in OpenCV® is an example of a function that may be used for calculating the image moment of a contour. Generally, an image moment defines a weighted average of the image pixels' intensities (which in this case, is limited to the pixels within the contour) and can be used to determine a centroid associated with an object.

[0060] In such examples, the object pose detector **203** may be configured to detect a change in pose of the at least two non-luminous objects based on changes in orientation of a line joining the locations within the image representation of each object (relative to some default orientation). The object detector **202** may be configured to fit a line between the centre points (or regions) identified for each object and to track changes in orientation of the line joining the centre points of each object. The user input generator **204** may be configured to generate a directional input based on changes in the orientation of the line joining the centre points of each object. The line joining the centre-points may be treated in the same way as the linear representation described previously in relation to generating a steering or change in viewpoint command.

[0061] More generally, the object pose detector **203** is configured to detect changes in relative pose of each object, so that a corresponding user input can be generated based on the detected changes in pose.

[0062] FIG. **5** illustrates schematically an example of a video image **500** in which a user is holding two oranges **502**A, **502**B as a video games controller. In FIG. **5**, oranges **502**A and **502**B are shown as having respective centre points **504**A, **504**B. The respective centre points are joined by line **506**. The line **506** corresponds to the line that may be tracked in order to generate a directional input to a video game. The video game may correspond to a racing game, for example. For clarity, the user's fingers are not shown in FIG. **5**, although it will be appreciated that these will likely be visible and may affect the contour detection as well as any subsequent calculation of the corresponding centre points.

[0063] The area of the contours detected for each object and their relative position within the obtained images may be tracked in the same manners as described previously. Similarly, the user input generator **204** may be configured to generate different respective user inputs based on changes in area and position of at least one of the contours associated with the objects. For example, a detected forward movement of the orange in the user's right hand may correspond to an 'accelerate' input, whereas movement of the orange in the user's left and may correspond to a 'brake' input. Alternatively, moving both of the oranges forward or backward may correspond to an accelerate and brake command respectively.

[0064] Additionally, movement of one or both of the objects above or below a threshold height may be identified as corresponding to a pause command. For example, if one or both of the objects can no longer be detected in the obtained image, it may be assumed that the player is no longer engaging with the video game. Alternatively, in such a case, it may be that the object pose detector **203** reverts to tracking the single object that can still be detected in the obtained images. The pose-tracking of the single object may be performed as described previously in relation to a single-object controller.

[0065] For objects that are asymmetric in at least one axis, such as bananas, it may be more likely that the user's hand interrupts a complete outline of the object. Hence, the object pose detector **203** may be configured to detect at least two contours associated with each object. The object detector **202** may therefore be configured to identify the four largest contours in the obtained images (or filtered images) as corresponding to the objects being held by the user. Again, there may be a prior step of filtering colours not corresponding to the objects from the images and generating a binary mask of the filtered images.

[0066] The object detector **202** may be configured to fit a line to each contour (e.g. along its length). The line fitted to each contour may correspond to a linear representation of the contour e.g. a line through the centre of the contour (in x-y). Each object may be associated with two linear representations—e.g. a line for each uninterrupted segment of the object. The object detector **202** may be configured to interpolate between the two linear representations generated for each object, so as to generate a single linear representation for that object.

[0067] It will be appreciated that, in some examples, it may be that a single contour is detected for each object, e.g. where the hand does not cover the front face of the banana (relative to the camera capturing the images). In such examples, each object may be associated with a single linear representation.

[0068] The object pose detector **203** may be configured to detect a change in pose of the objects being held by the user based on a change in orientation of a line intersecting the centres of the linear representations generated for each object. An example of this is shown in FIG. **6**, which schematically illustrates a video image **600** of two bananas being used to play a video game. In FIG. **6**, the contour of each banana is shown as contours **602**A, **602**B, having corresponding centre points **604**A, **604**B, joined by line **606**. Rotation of the line **606** joining the centre points **604**A, **604**B of each linear representation may be provided as an input to the user input generator **204**, which generates a corresponding directional input. The directional input may correspond to a steering or change in viewpoint input, as described previously. It will be appreciated that in FIG. **6**, the user's fingers are not shown for reasons of clarity.

[0069] It will be appreciated that, in embodiments where machine learning is used to detect the objects in the obtained images (e.g. via a cascade classifier), it may not be necessary to perform contour detection on the objects. In such embodiments, contour detection may be implicit from the localization of the object performed by the trained machine learning